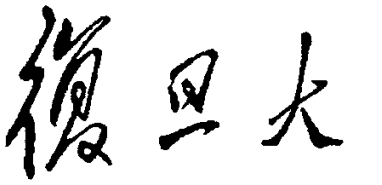


本科毕业论文

论文题目：	一种适用于移动自组织网络的基于分布式哈希表的服务发现协议
院　系：	计算机科学技术学院
专　业：	信息安全
姓　名：	葛一凡
学　号：	08300240120
指导教师：	孙未未

2012 年 6 月 15 日

摘要

分布式哈希表（Distributed Hash Table, DHT）是在对等（peer-to-peer, P2P）网络中进行资源查询定位的一种高效的方法。由于移动自组织网络（Mobile Ad hoc Network, MANET）和对等网络有许多相似的特性，例如网络中的节点自我设置并且是非中心的结构。因此本文认为，作为资源发现的一种，移动自组织网络中的服务发现可以借助分布式哈希表来实现。然而，由于分布式哈希表是被设计用于基于 Internet 的对等网络的，在带宽敏感的移动自组织网络中部署分布式哈希表并不是一个简单的任务。

在这篇论文中，本文基于分布式哈希表的思想，提出了一种适用于移动自组织网络的服务发现协议。该协议在很少地引入网络负载的情况下，将分布式哈希表与移动自组织网络高效地结合。分布式哈希表被用于辅助定位服务提供节点并且采用了一种分簇协议来抑制服务发现引入的控制数据包。本文在无线网络模拟器中实现了该协议，实验结果证明本文的协议是高效、可靠的并且仅会产生很少的网络流量，对网络压力很小。

关键词：移动自组织网络，服务发现，分布式哈希表

Abstract

Distributed Hash Table (DHT) has proven to be an efficientway of locating resources in peer-to-peer networks. As mobile ad hoc networks (MANETs) and peer-to-peer networks share similar properties such as self-configuring and decentralized, the service discovery in MANETs can also potentially benefit from the deployment of a DHT overlay. However, as DHTs are intended to be used in Internet based peer-to-peer networks, it is not appropriate to simply deploy such an overlay in the bandwidth sensitive MANETs.

In this paper, we propose a DHT-based lightweight service discovery protocol for MANETs, which efficiently implements a DHT overlay in MANETs with negligible control messages. The DHT substrate is introduced to help the locating of service providers and a clustering protocol is adopted to suppress the amount of control packets generated by service discovery. Our simulations conducted in the wireless network simulator also prove our protocol is swift, efficient and reliable as well.

Keywords: Mobile Ad hoc Networks, service discovery, distributed hash table

目录

图表目录

第一章　引言

移动自组织网络（Mobile Ad hoc Network, MANET）是一种多跳、节点自我配置的无线网络。在移动自组织网络的环境中，节点间的通信没有固定的网络基础设施的支持，并且网络中的节点都为其他节点的数据包进行路由转发。移动自组织网络一般由一组处于同一区域的具有无线通信能力的设备构成，图 1 展示了一个无线自组织网络的示例。由于移动自组织网络无基础设施以及无中心的特点，网络的构建十分灵活，可以被应用于战场通信、灾难救援等难以架设网络基础设施并且网络拓扑结构多变的场景之下，并且适合节点之间进行协同工作以及普适计算，本文认为移动自组织网络是部署面向服务架构的一个有研究潜力的应用场景。

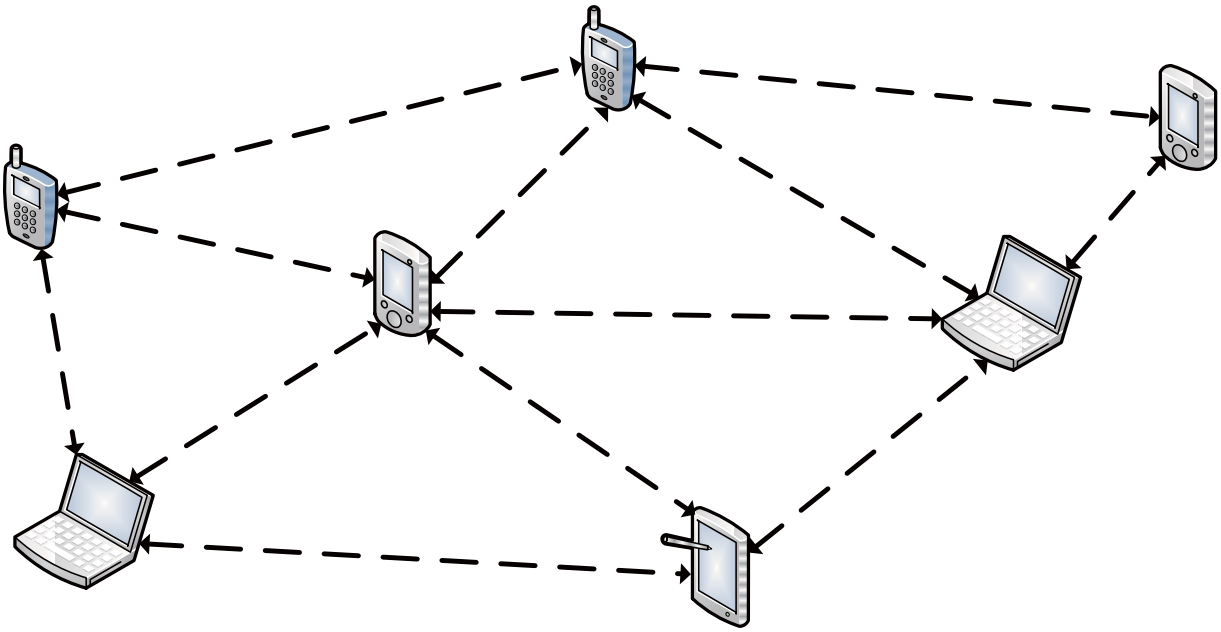


图 1：移动自组织网络

面向服务的架构（Service Oriented Architecture, SOA）是将系统的功能分割成独立单元的一组软件系统设计准则。面向服务的架构把应用所所提供的功能分解为独立的单元，这些功能单元被称为服务。用户可以通过这些服务的定义良好的接口来调用这些服务。面向服务的架构能够增强系统的灵活性，这对于在网络中部署系统的需求是十分重要的。

在面向服务架构中，服务的可发现性是一个需要得到保证的基础要求。服务发现协议的作用就是使网络中的设备可以查询和定位所需要的服务，并且向网络中的其他节点发布本节点所能够提供的服务。目前已经存在许多成熟的服务发现协议，如 Web Services

Dynamic Discovery（WS-Discovery）协议[14]、Universal Description Discovery and Integration（UDDI）[23]协议、Service Location Protocol（SLP）[20]协议等。然而，这些服务发现协议都是为在 Internet 下进行服务发现而设计的，本文所研究的移动自组织网络环境下的服务发现协议则需要与这些协议分开对待。

移动自组织网络环境中的服务发现与 Internet 中的服务发现存在很大的不同，这是因为相比 Internet，移动自组织网络有很多特殊之处：

- 移动自组织网络中的节点可以自由地移动，随意地加入或者离开网络。这使得网络的拓扑结构处于持续的变化之中。
- 无线信道是不可靠的，在数据传输的过程中经常发生干扰、碰撞等问题。
- 移动节点的资源（如处理能力、存储空间、电池电量等）有限。

这些特性使得无法在移动自组织网络中难以部署服务管理节点或者是注册服务器等设施，因为这样的中心节点设置会使网络容易出现单点故障。而多变的网络拓扑结构也使得需要较多的数据交换保持信息的一致性。因此为了适应移动自组织网络，需要一种无中心分布式的策略来进行服务发现。

对等（peer-to-peer, P2P）网络则是一种允许网络中的同时担任服务器和客户端的网络架构。在这样的网络架构中，节点处于平等的地位，不需要一个中心服务器去协调和维护网络。对等网络中节点的平等性和移动自组织网络中移动节点的平等性有很大的相似性。在这两者之间还有更多的共同点：

- 对等网络和无线自组织网络都是自我设置和无中心的。不需要中心服务器来协调网络。
- 对等网络中的对等节点和移动自组织网络中的移动节点都可以自由地加入和离开网络，因此这两种网络的拓扑结构都是频繁变化的。
- 在对等节点和移动节点之间的连接和数据交换是逐跳（hop-by-hop）的。尽管在对等网络中每一跳的传输一般是通过 TCP 连接进行而且一般对应着下层实际网络的多跳，而移动自组织网络中的每一跳传输都是直接通过 UDP 连接进行。

Napster[13]是最早出现的 P2P 应用。Napster 出现于 1999 年，是一个互联网音乐分享系统。而直到至今，资源分享仍然是最主要的 P2P 应用。而且从某种角度来说，服务发现也可以看成资源分享的一种形式。并且由于在这两种网络间存在着前面所提到的相似性，本文认为在对等网络中采用的一些技术和方法也同样可以用于移动自组织网络。当然，在对等网络研究中的的所需要考虑的问题也对移动自组织网络的研究中也需要考虑。

大多数的构建在对等网络上的系统使用分布式哈希表（Distribute Hash Table, DHT）来提供资源索引服务。在对等网络中，每一个资源或者服务都会被用一个哈希算法映射到

一个键值，而每一个对等节点都会负责存储一个特定的键值范围的资源信息。这种方法称为一致性哈希（Consistent Hashing）。一致性哈希算法能够起到平衡负载的作用，对等网络中的每一个节点所负责存储的键值量都是大致相等的。使用分布式哈希表，对等网络中的资源索引可以高效地进行。例如在一个由N个对等节点组成的对等网络中，资源查询请求可以通过$O(\log N)$跳的传递到达目标节点。

本文认为，利用这两种网络的相似性来设计一个高效的适用于移动自组织网络环境的服务发现协议是一个可行的研究方向。然而，只是简单地将传统的分布式哈希表与移动自组织网络相结合并不能达到令人满意的结果。这是因为尽管这两种网络之间存在着相似性，它们之间的差异也是不能忽视的。

要将对等网络结构应用在移动自组织网络中的一个首要的挑战就是对等网络一般是被设计工作在 Internet 下的，网络环境相对较为可靠，而移动自组织网络中的移动节点通过无线信道传送数据。由于分布式哈希表需要对等节点持续监视对等网络上的邻居节点，移动自组织网络中节点的高移动性以及有限带宽会使的分布式哈希表结构的一致性的维护耗费太高的代价。

因此，为了在移动自组织网络中适用分布式哈希表，需要针对移动自组织网络的特点进行调整和优化。由于无线网络的特性，在移动自组织网络中，节点之间的数据交换都是通过 MAC 层广播的形式进行的，即便是单播也是如此，这和有线网络中的情形很不一样。因此，当一个节点发送一个数据包时，所有在这个节点的广播传输范围覆盖内的节点都会收到这个数据包的一个副本，收到数据包的节点并会根据实际情况转发或者丢弃这个数据包。那么，如果节点能够在将不需要的数据包丢弃之前将其中有价值的信息缓存在本地，就可以有效地减少网络的压力，并且提高路由发现和服务发现的效率。这种方法被称为“偷听”（overhearing）。

另一个可以用来提高性能的手段是将移动自组织网络中的节点划分到不同的“子网”中。用这样的方法，本文可以维持相对稳定的网络拓扑结构并减少在网络上传播的数据包数量。一种常用的用来动态地在移动自组织网络中划分“子网”的方法是分簇（clustering），通过给移动节点分配不同的分簇角色（如簇头节点、网关节点、普通节点等），使得节点之间形成簇的结构。然而，如果在移动自组织网络中节点持续移动，分簇的变化也会十分频繁，为了维护分簇的结构则需要较多的数据交换，因此过于复杂的分簇协议将会引入过多的的网络负载，使得由分簇带来的效率提高被抵消。所以，在设计分簇协议时必须做好权衡。

总结来说，本文所期望的服务发现协议由一个用于定位服务的轻量级的分布式哈希表、偷听机制以及一个最简化的按需式分簇协议所组成。而在路由层，本文使用反应式（reactive）路由协议来节省带宽资源。

在下一章，本文将介绍移动自组织网络中的服务发现、分布式哈希表以及分簇协议的相关工作和一些背景知识。

在第 3 章，本文将介绍所提出的服务发现协议，并主要讨论基于分布式哈希表的服务发布和服务发现过程。

在第 4 章，本文使用了 Passive Clustering 协议对所提出的服务发现协议进行了优化，在不增加额外的控制数据包的前提下构建出簇的结构以减少不必要的数据洪泛。

在本文的第 5 章，本文介绍了仿真实验的配置以及实验结果，并对实验结果进行了分析和解释。并在最后对论文进行了总结。

第二章　背景及相关工作

2.1. 服务发现方法

目前在服务发现领域已经有了大量的成熟的研究成果。研究者们提出许多适用于不同应用场景的服务发现协议。Jini[1]是一个基于 Java 扩展的用于建立分布式系统的架构。Jini 提供了服务发布和服务发现的功能使得用户可以在网络中调用所需要的资源。Universal Service Discovery and Integration (UDDI)[23]是一个服务描述标准，它使服务的提供者可以在服务目录中注册服务信息，并使服务能够被发现和调用。然而，这些已有的工作都不适合部署在移动自组织网络的环境下。举例来说，UDDI 协议采用了一个公共可见的中心 UDDI 节点来提供服务发现的支持。如果需要服务可被返现，服务的提供者需要将服务信息提供给这个中心 UDDI 节点，而服务的使用者也是需要通过中心 UDDI 节点获取服务的描述信息。因此，服务的能否被发现取决于这个中心目录是否可以被访问到。由于在移动自组织网络里这个中心目录将会使节点承担过多的处理工作，单点故障的概率大幅提高，而成为整个网络的瓶颈。所以这些方法是不适合于移动自组织网络之类的分布式环境的。

因此，无中心的服务发现协议将会更适合于移动自组织网络的要求。移动自组织网络中已有的服务发现协议大体有两种主要思路，一种是服务的提供节点采用“服务广告”的机制，将服务的信息推送给服务的潜在用户；而另一种思路是客户节点在需要发现服务的时候去网络中查询服务信息，但是由于网络的无中心特性，这种查询往往是通过基于洪泛的方式进行的。Konark[6]和 Allia[19]是典型的用于移动自组织网络的服务发现协议。Allia 采用了对等缓存的机制来进行服务发现，网络被按照位置被划分成小的称为 Alliance 的单元，服务的提供者将它们的服务信息在邻近区域内进行广告。当服务发现请求被发起时，服务的请求者将会首先在其所在的 Alliance 中进行查询；如果失败，那么将会广播或者多播该服务请求。

Konark 是一个服务发现中间层协议。在 Konark 中，采用了以上所述的两种思路的结合，服务的客户端节点可以主动地进行服务发现以及被动地缓存服务广告的数据，并使用了一个树结构的服务目录被用来维护和管理服务信息。

文献[31]中，作者针对移动自组织网络的特点提出了一种基于服务广告的服务发现方法，并将服务的质量考虑在内，根据服务的可靠程度调整服务广告的范围。

2.2. 分布式哈希表

分布式哈希表是广泛应用于对等网络中的用于提供资源查询的技术。目前成熟的分布式哈希表系统已经有很多种，Chord[24]、Content Addressable Network (CAN)[18]、Pastry[22]、Tapestry[30]以及 Kademlia[12]都是现在常见的分布式哈希表系统实现。在这些实现中，对等网络中的每一个资源以及对等节点都使用同一个哈希算法（如 SHA-1、MD5）来获得一个键值。由于哈希算法的特性，这些被分配的键值在键值空间（key space）上是基本均匀分布的。这些键值使用一致性哈希分配给网络中的对等节点。一致性哈希可以很好地应对系统中对等节点个数改变的情形。在一般的哈希算法中，如果槽（slot）的个数改变，将会造成已有槽中的数据的重新分配，而一致性哈希可以尽可能地减少需要重分配的数量。若K是键值的总数而n是槽的总数，那么平均会有K/n个键值被重新分配。举例来说，在 Chord 中，所有的键值都被排列在一个环形的键值空间上，键值k被自身键值最为接近k的对等节点所管理。当有一个对等节点加入或者离开网络时，槽的数量（对等节点的数量）发生了变化，此时需要调整的键值只有在环形键值空间上这个加入或者离开网络的节点的键值邻近的键值。这个特性对于分布式系统，尤其是移动自组织网络十分重要，因为在移动自组织网络中节点的移动频繁，常常会有节点加入或者离开网络。由于键值的重分配需要节点之间的数据交换，如果涉及到过多的节点，将是很耗费资源的。

Pastry 和 Tapestry 使用了类似的基于“面向前缀路由”（prefix oriented routing）的分布式哈希表设计。面向前缀路由[16]是用 Plaxton 等人在实际的对等网络系统出现之前发明的，其主要思想在于，每一个资源的键值都会被映射到和该键值有最长的公共前缀的键值所对应的对等节点。使用这样的技术，可以做到数据的均匀分布，并且在有N个对等节点的系统中，路由可以在$O(\log N)$跳内完成。

然而，由于移动节点的高移动性以及处理能力、电池电量和带宽的限制，要把这样的对等网络结构应用在移动自组织网络中并不容易。目前在这一方面已经有了一些尝试。文献[7]中，作者讨论了对等网络与移动自组织网络的相似性以及如何利用这种相似性设计一个适用于移动自组织网络的分布式哈希表系统。文中提出的分布式哈希表系统基于 Pastry，并进行了优化和调整。Etka[17]是一个将 Pastry 和 Dynamic Source Routing (DSR)[8]路由协议相整合的跨层协议设计。在文献中，作者经过仿真实验验证了协议的效率，实验结果说明了基于分布式哈希表的协议比基于物理层广播的协议更能高效地支持应用。MADPastry[28]是另一个适用于移动自组织网络的分布式哈希表系统。与 Etka 不同，MADPastry 使用了自组织网络按需平面距离矢量路由协议（Ad hoc On-demand Distance

Vector, AODV）[5]作为其路由协议。在路由协议之上是一个简化的 Pastry 结构来更好地适应移动自组织网络的环境。MADPastry 协议更好地将节点的位置考虑在内，协议在键值空间上选取了一些 landmark 键值，这些 landmark 键值将键值空间划分成均匀的区域。键值在键值空间上最接近 landmark 键值的节点被选中作为 landmark 节点。这些 landmark 节点周期性地发送数据包在邻近区域内标明自己的存在。收到这个数据包的节点将成为这个 landmark 节点为簇头的簇内节点。

在文献[26]中，作者提出了一种更复杂的系统结构。文中提出了一种基于分布式哈希表的对等结构的大规模多人在线游戏（Massively Multi-player Online Game, MMOG）的设计。作者采用了 Random Landmarking[25]的方法来创建节点分簇，并且将分簇以类似于 Hierarchical State Routing （HSR）[15]的方法构建成树状结构。每一个节点维护两种分布式哈希表路由表，远端分布式哈希表路由表存储的是每一个分簇的 landmark 键值，而本地分布式哈希表路由表被组织成虚拟环路由表[3]的形式来支持簇内查询。这种方法尽管能够有效地适应移动自组织网络的动态环境，但是协议太过复杂，维护成本过高。

2.3. 分簇协议

文献[27]是一个移动自组织网络中分簇协议的综述，在文中，作者讨论了移动自组织网络中节点分簇的意义并对移动自组织网络中常用的分簇协议进行了分类并对其效率进行了评估。简单来看，移动自组织网络的分簇协议大体可以分为两种，主动式（proactive）分簇协议和反应式（reactive）分簇协议。主动式分簇协议一般持续地尝试建立和维护分簇结构。文献[11][2]提出了一些主动式分簇的机制。这些协议主要关注的是簇头的选取以及节点加入和离开分簇的行为。主动式分簇协议的特点是控制报文持续传输即便当前并没有必要去维护一个分簇结构，因此存在被浪费的带宽资源。

而反应式分簇协议则与此不同。[4]提出了一个需求驱动的分簇机制。文章作者引入了一个邻近节点识别协议（neighborhood recognition protocol）并将其用于分簇的建立和维护。分簇的相关操作只有在节点有数据包需要发送的时候才会进行，因此为了维护分簇而引入的控制数据包是很少的。

文献[10]中提出了一种被动分簇协议（Passive Clustering protocol），这是又一种反应式分簇协议。被动分簇协议被用于达到移动自组织网络中的高效洪泛。在这个协议中，所有的分簇状态信息都附加在用户数据包中，因此不需要引入额外的控制数据包。分簇的结构是用户数据传输时顺带产生的。而本文所提出的方法在很大程度上收到这个被动分簇协议的启发。

第三章　服务发现协议设计

3.1. 概览

在本文中，由于本文的重点是服务发现协议的设计，因此主要关注的是在服务发现中节点的行为以及节点如何与网络下层交互，并对其性能进行评估。而对于服务属性和类型的描述格式，本文不会做太多关注。在本文中，本文采用了文献[9]中提出的关于服务和服务发现的定义：

- **服务**指的是能够被移动节点所利用的硬件或软件功能。
- **服务器**指能够为其他节点提供至少一个服务的网络中的任何移动节点。
- **客户端**指希望使用某一在移动自组织网络中的服务器提供的服务的任何移动节点。
- **服务发现**是从服务描述到服务器的 IP 地址的映射。

图 2 展示了本文提出的服务发现协议的架构。架构分成三层：应用层、服务管理层和网络层。

应用层上工作的是服务器所提供可供其他移动节点所使用的服务，如打印服务、网络存储等等。

而本文提出的服务发现协议的核心功能在服务管理层上实现。若服务器需要再网络中发布某个服务，将调用服务发布模块（Service Publish module），启动服务发布过程。当节点作为客户端希望查询被用户所请求的服务时，则需要调用服务发现模块（Service Discovery module）以启动服务发现过程。每一个节点在本地维护一个服务缓存（service cache），服务缓存中存储邻近节点是否存在的信息以及其所提供的服务的情况。这个服务缓存由图中的 Service Cache Manager 进行管理，并将会在服务发现过程中帮助节点更快地发现所需要的服务。图中的 Forwarding Manager 负责管理节点对数据包的发送、转发或者其他的处理。当节点收到一个数据包，比如一个包含有服务信息的服务发现响应包时，Forwarding Manager 将会首先将解析数据包，并通过 Service Cache Manager 更新本节点的服务缓存，还需要判断节点下一步是需要将该数据包转发还是直接丢弃，如果需要将数据包转发，则会使用分布式哈希表路由表（DHT Routing Table）来帮助找到转发的下一跳节点。

在本协议的网络层中，本文使用了知名的自组织网络按需平面距离矢量路由协议（Ad hoc On-demand Distance Vector, AODV）作为下层的支持服务发现系统的路由协议。

由于 AODV 协议是一个响应式路由协议，在没有路由发现的需要时不会产生额外的网络流量，因此在维护路由表时将会产生较小的网络负载。

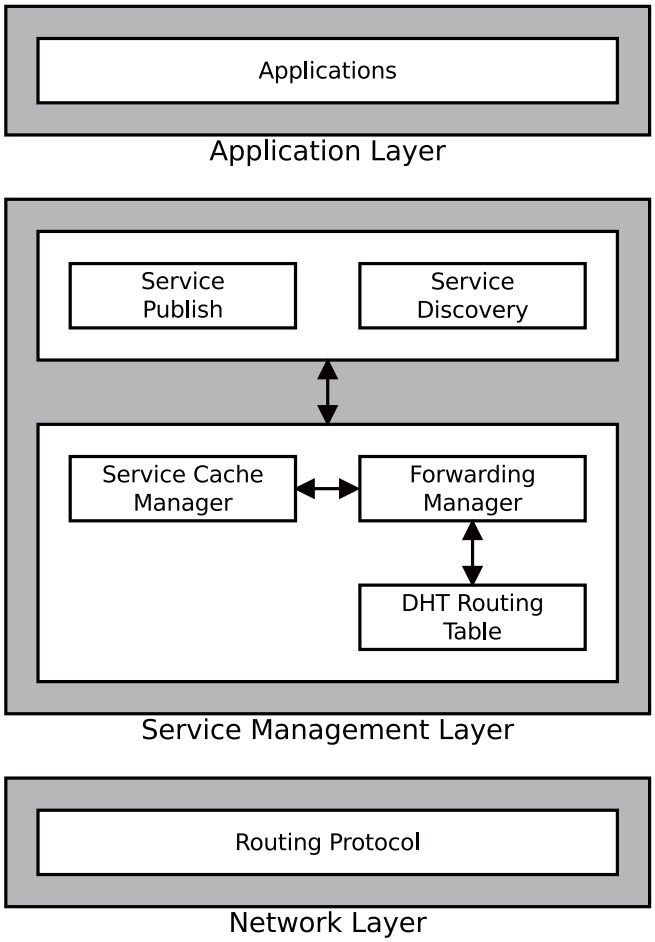


图 2：服务发现协议结构

3.2. 基于分布式哈希表的服务发现协议

本文提出的协议利用了一个类似于 Tapestry 的分布式哈希表来支持服务发布和发现功能。分布式哈希表用于为节点提供所请求服务的路由信息。

在网络中，每一个被服务器提供的服务都有一个服务描述（SD）。使用一个哈希算法H（实际操作是可以是 SHA-1、MD5 等哈希算法），本文从每一个SD生成一个键值K_S。

同时，在网络中的每一个节点都会有一个唯一的节点 ID，在实际的网络情况中，这个节点 ID 可以是节点的 IP 地址或者是 MAC 地址，在这里用N_{id}来表示。用和前面生成

K_S所用的相同的哈希算法从每一个N_{id}生成键值K_N。由于在生成K_S和生成K_N时使用的是同一个哈希算法，因此两者的输出（K_S和K_N）是在同一个键值空间上的。

本文提出的服务发现协议提供的两个最基础的功能如下：

1. PUBLISHSERVICE (SD, N_{id})：节点N_{id}发布一个服务SD并使其在网络中可被调用。节点N_{id}作为服务SD的服务器节点。
2. DISCOVERSERVICE (SD, N_{id})：节点N_{id}发起一个服务发现以获取提供服务SD的服务器的信息（IP 地址，跳数等）。节点N_{id}是客户端节点。

本文接下来将主要介绍这两个功能模块的设计和实现。

3.2.1. 服务发布

服务发布的操作与 Tapestry 中的相类似，每一个在网络中的服务都会将其键值K_S映射到一个网络中的节点，这个节点被称为K_S的根节点（root）。在所有的节点中，根节点的键值K_N应是最接近K_S的。但是由于在移动自组织网络的环境中，节点对网络中其他节点没有全局的认识，因此没有办法直接从键值空间中挑选出根节点。所以，根节点的选择是在服务发布的过程中逐步进行的。

当一个服务器需要发布一个服务时，将会创建一个 SERVICE PUBLISH 包，其中包括了本服务器的 IP 地址以及需要发布的服务的键值K_S。转发这个 SERVICE PUBLISH 包的节点将会倾向于选择一个键值K_N比本节点的键值更接近K_S的节点作为数据包转发的目标节点。具体来说，选择的目标节点将会是键值比当前节点的键值与K_S有更长的公共前缀位的节点。

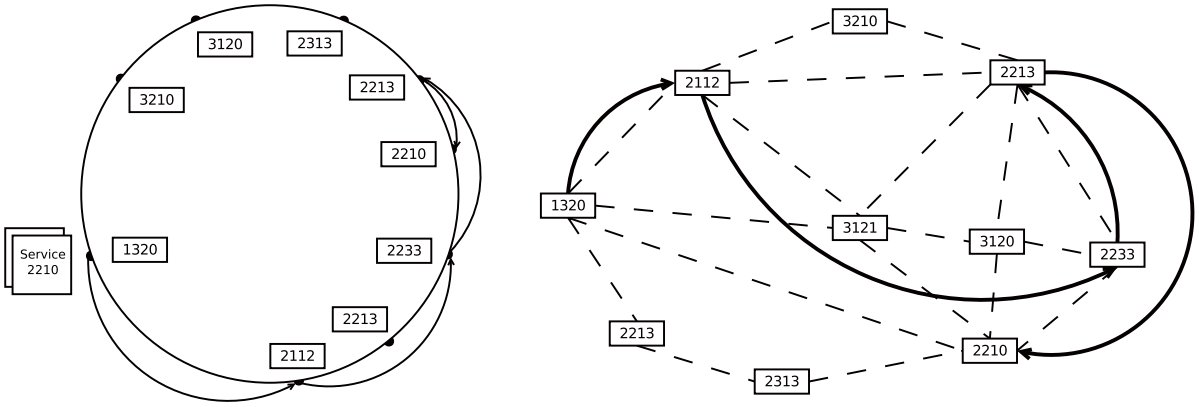


图 3：基于分布式哈希表的服务发布

图 3 展示了一个服务发布过程的实例。服务器（K_N=1320）需要发布一个服务（K_S=2210）。使用面向前缀的路由方式，每一个节点都会将数据包发送给键值K_N和K_S（2210）

有更长的公共前缀的节点。因此，如图 2 所示，最终选择路径 1320⇒2112⇒2233⇒2213⇒2210 直至数据包到达了根节点。而在这个 SERVICE PUBLISH 包转发的过程中，所经过的节点将会把其中的服务信息缓存在本地的服务缓存中，以便提高之后的服务发现效率。

这种面向前缀的路由方式类似于无类别域间路由（Classless Inter-Domain Routing, CIDR）[21]，但是一个很大的不同之处在于，和无类别域间路由中 IP 地址的子网划分不同，本方法中节点的键值并没有位置距离上的意义。在图 2 中可以看到，即便是中间节点的键值逐步接近了根节点的键值，实际的到根节点的距离并不是总是减小的（但是总体上看，还是逐渐接近根节点的）。另一个需要注意的细节是在，在分布式哈希表中的一跳往往包含有实际的物理网络的多跳。

为了支持这种面向前缀路由，每一个节点都维护一个分布式哈希表路由表。路由表的行数和列数是由哈希算法的输出的位长度，以及所选择的底数（base）所决定的。在此本文假定哈希算法的输出的位长度是L，而底数是2^{β}，那么分布式哈希表路由表就将会有$\lceil L/\beta \rceil$行，而每行有2^{β}项。在前面图 2 的例子中，所选择的$\beta = 2$因此底数2^{β}为 4，位长度L为 8。在这个路由表中，在第i行第j列的项是一个指向一个链表的指针，链表中存放的是键值和当前节点的键值有$i-1$个公共前缀数字并且第i个数字是$j-1$的节点。

使用分布式哈希表路由表，节点可以容易地找到其转发 SERVICE PUBLISH 包的下一跳目标节点。此处本文假定$lcp(K_1, K_2)$是K_1和K_2的最长公共前缀长度，并且K^i是K的第i位数字，那么在分布式哈希表路由表的第$lcp(K_N, K_S)+1$行，第$K_S^{lcp(K_N,K_S)+1}+1$列的项将会指向一个包含了比当前节点的K_N更接近K_S的节点。

图 4 中是分布式哈希表路由表的简单例子。图中所表现的是键值为 2112 的节点，即图 3 中第二跳节点的分布式哈希表。为了描述的方便，图中路由表中的项中填写的是节点的键值，在具体的实现中，路由表中的项是指向一个包含有结点信息的链表的指针，其中包括有节点的 IP 地址，节点键值等相关信息。从图 4 中可以看出路由表的结构，例如在第 1 行中所存储的键值域 2112 的公共前缀长度为 0，并且第一个前缀数字与列号-1 相同，如第 1 行第 2 列中的项的第一个前缀数字为 1。

当键值为 2112 的节点接收到服务 2210 的 SERVICE PUBLISH 包时，需要从分布式哈希表路由表中找到转发的下一跳节点。根据前面提到的步骤，在这个场景下$K_N = 2112$并且$K_S = 2210$，求得$lcp(K_N, K_S)+1 = 2$，$K_S^{lcp(K_N,K_S)+1}+1 = 3$，因此可以从路由表的第 2 行第 3 列的项中找到转发的下一跳节点的信息，从图 4 中可以找到第 2 行第 3 列中存储的是 2233，因此选择的下一跳节点就是键值为 2233 的节点。

当分布式哈希表路由表中的某一项指向链表中的节点数超过一个，比如图 4 中的第 1 行第 1 列的项，则需要从这个链表中选择一个最合适的作为下一跳节点。在此本文以距离

	(1)	(2)	(3)	(4)
(1)	**0**210 **0**120	**1**210	**2**012 **2**123	**3**130
(2)	<u>2</u>**0**10	<u>2</u>**1**20 <u>2</u>**1**31	<u>2</u>**2**33	<u>2</u>**3**30
(3)	<u>21</u>**0**0	<u>21</u>**1**3	----	<u>21</u>**3**3
(4)	----	<u>211</u>**1**	----	<u>211</u>**3**

图 4：键值为 2112 的节点的分布式哈希表路由表，图中的加下划线的数字是与当前节点的键值的公共前缀，加黑的数字是与分布式哈希表路由表的列号相关的数字

当前节点的跳数距离作为选择标准，这个跳数距离可以从节点的路由协议所维护的路由表中获取，并选择一个距离当前节点跳数距离较近的节点作为转发该数据包的下一跳节点。

在某些情况下，很有可能用上面的方法找到的路由表中的项指向着一个空的链表，比如图 4 中的第 3 行第 3 列的项。这时，转发数据包的下一跳节点将会选择为和公共前缀长度和本节点相同但是相较本节点更为接近K_S的节点。当在这两种情况下在路由表中都没有节点可供选择时，路由过程就会就此终止，当前的节点被视为所发布服务的根节点。所发布服务的信息，包括服务器的 IP 地址、服务描述等都会存储在根节点上。

3.2.2 服务发现

服务发现过程与服务发布过程是很相似的。当用户需要某一个服务时，用户所在的客户端节点将发起一个服务发现过程，此时客户端节点生成一个 SERVICE REQUEST 包，其中包含有所需要的服务的键值K_S以及客户端的 IP 地址。使用分布式哈希表路由表，这个 SERVICE REQUEST 包被转发至服务的根节点。转发的过程与服务发布时类似，都是根据键值的前缀匹配程度逐跳转发。由于本文采用了服务缓存的机制，很有可能在转发至根节点的路径上的中间节点就可以提供所需要的服务信息。如果这样的情形发生，那么当前的节点就将不再转发 SERVICE REQUEST 包，而转而初始化一个包含有服务信息的 SERVICE REPLY 包以响应该服务发现请求并将这个 SERVICE REPLY 包发送回客户端节点。

第四章　分簇协议

正如前面的章节所讨论的，本文的服务发现协议是采用了 AODV 作为路由协议。作为一个按需路由协议，当节点需要发送数据包但缺乏必要的路由信息时，进行路由发现过程是必不可少的。按照 AODV 路由协议的设计，当节点发起一个路由发现过程时，路由发现请求包将会在整个网络上洪泛，直到请求包被能够提供所需要的路由信息的节点所接收到，并将一个路由请求响应包回复给发起路由发现的节点。除此之外，正如前面一章所提到的，如果在服务发布或者服务发现时，分布式哈希表路由表不能提供有效的信息时，当前节点也会发起一次洪泛以完成服务发布或者服务发现过程。

然而，洪泛是会消耗大量的网络带宽资源的操作，而且如果有大量的节点在短时间内发起洪泛，甚至可能会使网络中充斥着大量的广播数据包以导致广播风暴使网络瘫痪。而从另一方面来看，如果网络中的节点比较稠密，节点的广播覆盖范围有较多重叠时，那么在进行洪泛时很多的转发都是不必要的，因此可以通过抑制这些不必要的转发来减少洪泛对网络的压力。所以，本文采用了类似于文献[10]中提出的 Passive Clustering 算法来优化本文的协议，以减少不必要的洪泛数据包转发。

在协议中，本文为网络中的节点分配有不同的分簇内角色，根据节点角色的不同，网络将被划分成动态的分簇。在网络中，节点的分簇内角色可能有：簇头节点、网关节点、普通节点以及初始节点，而这些角色状态是根据节点的行为以及节点收集到的邻居节点的行为所共同决定的。

初始节点时网络中节点的初始状态。刚加入到网络中的节点会被分配为初始节点，除此之外，当网络中的某一个节点在一段时间内处于非活跃状态，即没有发送数据包也没有接收数据包，这个节点也会被转换为初始节点状态。

簇头节点是一个分簇的领导节点，每一个分簇中只会有一个簇头节点存在，分簇中的其他节点都维护有一个簇头节点列表，其中存储有该节点所在分簇的簇头节点的信息。由于同一个节点可能处于多个分簇的交叉位置，因此簇头节点列表中存储的簇头节点信息常常有多个。这个簇头节点列表中的信息是作为“软状态”存储的，每一条簇头节点信息都被分配有一个定时器，协议将定期检查最近是否收到来自该簇头节点的数据包，如果在一定的时间间隔中没有收到，则将该簇头节点信息从列表中清除。

在协议中，只有初始节点才有可能成为分簇中的簇头节点。由于本文中采用的分簇协议不会引入新的控制数据包，节点只有在有数据包需要发送时才能够将节点分簇状态附加在数据包中告知其他节点。因此当一个初始节点收到数据包时，将转变为一个特殊的簇头

就绪状态，这是一个不会附加在数据包中告知其他节点的内部状态，当这个节点有数据包需要发送时，其将会在此时转变为簇头节点并将这个状态附加在将要发送的数据包中，把这个簇头节点信息通知给邻近的节点。

当一个节点同时能够收到来自两个不同簇头节点的数据包时，说明这个节点位于这两个簇头节点所领导的两个分簇的交叉位置。由于这个节点所处位置的特殊性，可以起到在这两个分簇之间转发数据包的作用。因此，这个节点的状态将转变为网关节点。而在其他的情况下，如果一个节点不是簇头节点，不是网关节点，也不是初始节点，那么这个节点将作为一个分簇中的普通成员，转变其状态为普通节点。通常情况下，一个只能收到来自一个簇头节点数据包的节点会是一个普通节点。分簇协议对于减少网络洪泛数据包的贡献在于，当普通节点接收到洪泛数据包时，不会对其进行转发，在簇间进行转发的任务交给网关节点，而在簇内公告数据的任务交给簇头节点。

在实际情况下，如果节点的分布比较密集，有很大可能性一个节点会处在多个簇头节点的广播范围之内，因此可能会出现网络中除了簇头节点，其他大部分节点的状态都是网关节点的情况，这样分簇对于网络洪泛数据包抑制的作用就不很明显。所以约定当一个网关节点的簇头节点列表中保存的簇头节点数量少于这个节点所了解到的网关节点的数量时，当前的网关节点将会转变为一个普通节点。为了做到这一点，每一个节点还需要维护一个类似于簇头节点列表的网关节点列表来保存该节点所能接收到的网关节点的相关信息。同样的，这个网关节点列表中的记录是以“软状态”存储，每一条记录都分配有一个计时器，如果在一个时间段内没有受到来自该条记录中的网关节点的数据，则将其从列表中去除。

正如前面所提到的，节点的角色状态信息是附加在节点所发送出的数据包的 MAC 层包头中的，其他接收到或者“偷听”到这个数据包的节点将会先从中获取到这个角色状态信息，然后再按照路由协议的逻辑处理这个数据包，包括转发或者丢弃等处理。每一个节点的状态都不是保持不变的，而也是作为“软状态”存储，以此来适应移动自组织网络动态的拓扑结构。每个节点都会设置一个定时器来监视节点的非活跃时间，并在必要的时候将节点的状态转变回初始节点状态。

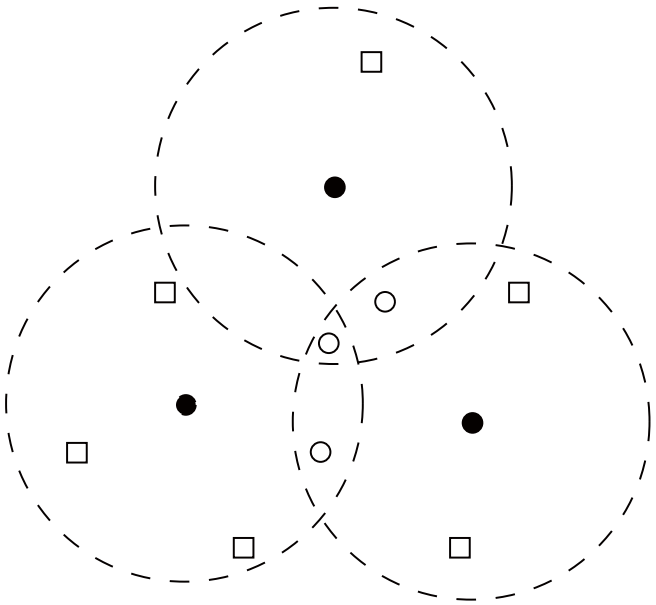


图 5：网络分簇示意

图 5 是一个网络中分簇结构的示意图。在图中，3 个黑色点代表的是簇头节点，虚线圈则分别是这三个簇头节点的广播传输覆盖范围，也代表了三个分簇的范围。图中白色的点处于两个分簇的交叉区域，是两个分簇之间的网关节点，而剩下的方形节点则是普通节点。当黑色点的簇头节点发送一个广播数据包时，在其为圆心的虚线圈中的节点都会接收到这个广播数据包，但是根据分簇协议的逻辑，只有白色点的网关节点会将这个广播数据包继续广播以传递给其他分簇中的节点，当普通节点收到广播的数据包时，将会将其丢弃而不会参与到洪泛的过程中去。因此，使用这种分簇协议将能够在不引入额外的控制数据包的前提下构建和维护分簇的结构，并能够减少不必要的洪泛数据包转发，减轻对网络的压力。

第五章　仿真实验

5.1. 实验配置

本文在无线网络模拟器 GloMoSim[29]上实现了本文的服务发现协议。在实验过程中所选取的一些重要参数如表 1 所示。

表 1：实验参数设置

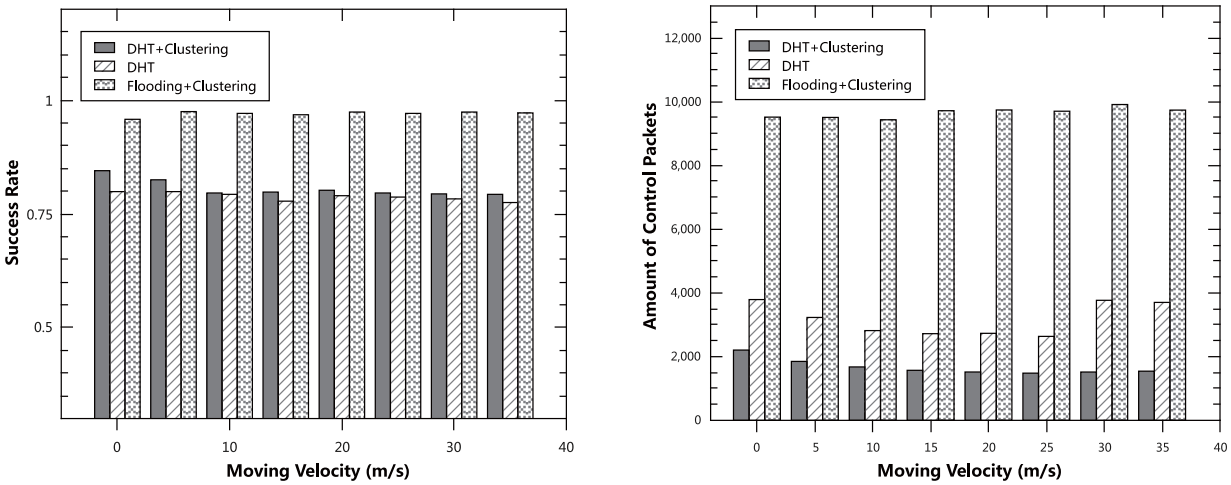
参数名	值
实验时间	15 分钟
模拟区域面积	1000 米*1000 米
节点放置方式	随机
节点移动模式	随机路点

在实现本文的服务发现协议的过程中，为了提高协议的效率，采用了跨层的设计方式，将分布式哈希表以及服务发现的功能在路由层进行了实现。

在这一章中，本文通过比较三种不同的服务发现协议实现来评估了本文的实验发现协议的性能。三种服务发现协议分别为：同时使用了分布式哈希表和分簇的协议、使用了分布式哈希表但是没有使用分簇的协议以及使用洪泛服务请求进行服务发现和分簇抑制洪泛数据包的协议，并且每个节点都会维护服务缓存。这三种方法在下文中分别用 *DHT+Clustering*、*DHT* 以及 *Flooding+Clustering* 来表示。

5.2. 实验结果

本文首先研究了不同的节点移动速度对于服务发现协议的服务发现成功率（success rate）的影响。服务发现成功率是在总共的服务发现请求尝试中，最终查询到需要的服务的比率。这项实验的结果体现在图 6(a)中，节点的移动速度由 0 m/s 变化到 35 m/s，从中可以看出在三种不同的服务发现协议中，尽管 *Flooding+Clustering* 是最直接的方法，但是效果却是很好的，服务发现成功率最高，达到了 90%以上的服务发现成功率。这是由于每一次需要进行服务发现时，客户端节点都会将服务请求包 SERVICE REQUEST 进行洪泛，因此近乎整个网络都会参与到服务发现过程中，这就保证了服务发现的高成功率，而且网络中的节点会缓存服务信息，因此几乎所有被请求的服务都会最终转发到服务的提供节点或者存储有相关服务信息的节点，使得服务可以被查询到。但是高成功率带来的代价是这



(a) (b)

图 6：节点移动速度对于服务发现成功率及控制包数量的影响

种基于洪泛的方法需要大量的的控制数据包，导致对于网络的压力很大，这一点可以在图6(b)中看出。与此相比，由于基于分布式哈希表的两种方法采用“尽力而为”的策略将服务请求 SERVICE REQUEST 发送给服务的提供节点，其成功率会比基于洪泛的方法略低，但是仍能保持在 80%，处于可以接受的范围，并且由于所消耗的网络资源较低，这个缺点可以通过重新发起服务发现来弥补。而 *DHT+Clustering* 和 *DHT* 相比，*DHT+Clustering* 的成功率略高，但差距不是很大，这是因为分簇主要影响的控制包数量，而当网络压力较小时，服务发现成功的几率会更大。

在图 6(b)中是控制数据包的数量随节点移动速度的变化。与前一组实验一样，节点的移动速度变化范围仍然是 5 m/s 至 35 m/s，控制数据包包括有服务发现请求的 SERVICE REQUEST 包、服务请求响应的 SERVICE REPLY 以及服务发布的 SERVICE PUBLISH，这些控制数据包的大小都是近似的，因此本文使用控制数据包的数量来衡量服务发现协议给网络带来的压力。正如前面提到的，虽然 *Flooding+Clustering* 的服务发现成功率较高，但是其给网络带来的压力远大于另外两种基于分布式哈希表的方法。从图 6(b)中可以看出，*Flooding+Clustering* 的控制数据包数量大约是 *DHT+Clustering* 的 5 倍，大约是 *DHT* 的 3-4 倍，从中可以看到分布式哈希表对于网络负载的控制有着明显的作用。而将 *DHT+Clustering* 和 *DHT* 进行比较，*DHT+Clustering* 的网络负载相较 *DHT* 也有很大的优势，控制包数量大约是 *DHT* 的二分之一，从中可以看出分簇对于网络流量也有很好的抑制作用。

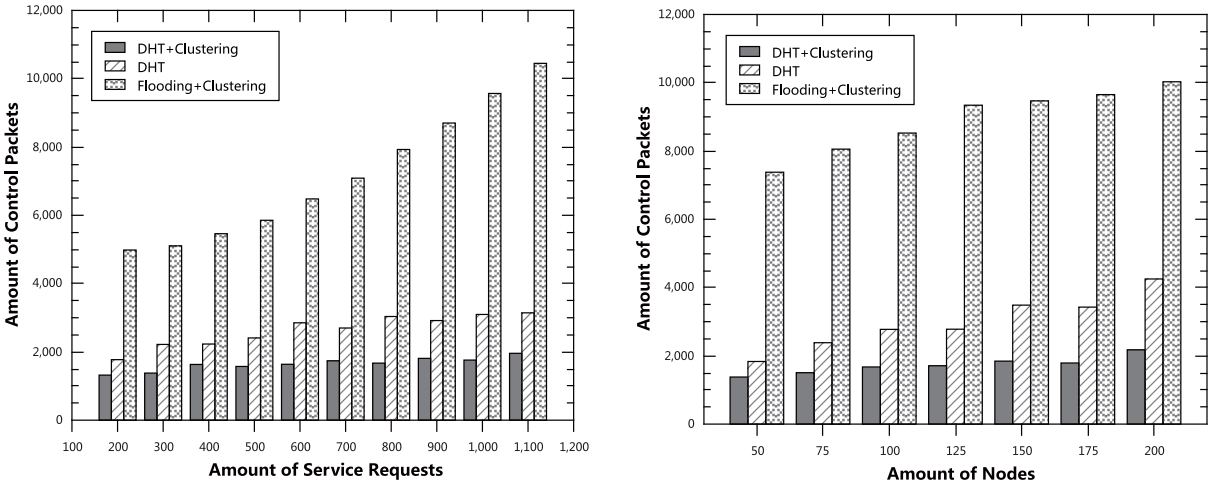


图 7：服务发现请求数量及节点数量对于控制包数量的影响

同时，本文还研究了其他因素对于网络中引入的控制数据包数量的影响。实验的结果如图 7 所示。图 7(a)中展示了服务请求的数量对于控制数据包数量的影响。在这一组实验中，服务的请求量从 200 个请求变化到 1100 个请求，从图中可以观察到明显的变化趋势。*Flooding+Clustering* 的控制数据包量有很显著的增长，而其他两种协议的增长则不是很明显，这是由于服务请求量的增加就意味着发送的 SERVICE REQUEST 包增加，而这个增加量在 *Flooding+Clustering* 中尤其明显。而在图 7(b)中，可以看到网络中节点的数量对于控制数据包数量的影响。在这一组实验中，节点的数量由 50 增长到 200，而模拟场地面积总

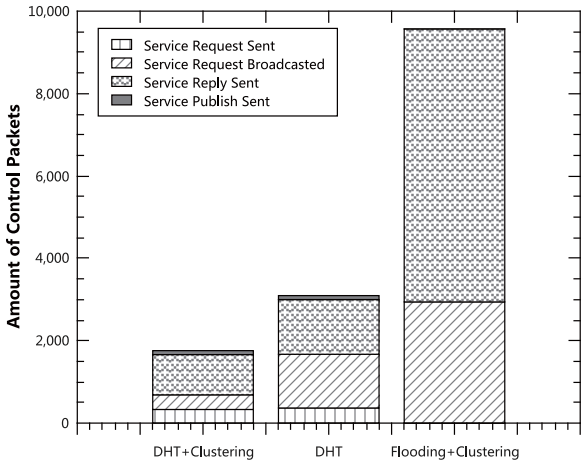


图 8：控制数据包的组成

大小是不变的 1000 m*1000 m。因为节点数增加意味着节点的密度增加，每发起一次洪泛，参与到转发洪泛数据包的节点都增加了，因此 *Flooding+Clustering* 以及 *DHT* 的控制数据包数量都有明显上升。而对于 *DHT+Clustering*，洪泛数据包被最大限度地抑制，因此控制数据包总量的上升很小。

在图 8 中，本文对三种协议中控制数据包的组成进行了分析。从中本文可以清楚的观察到，*DHT+Clustering* 中广播的 SERVICE REQUEST 包远比在 *DHT* 中的少，可以看出分簇对于洪泛数据包的抑制是十分明显的。而将 *DHT+Clustering* 和 *Flooding+Clustering* 进行比较，无论是 SERVICE REQUEST 还是 SERVICE REPLY，*DHT+Clustering* 都要比 *Flooding+Clustering* 少很多，从中可以看出分布式哈希表在减轻网络负载方面起到的重要作用。

第六章　结论及未来工作

在本文中，我们提出了一种适用于移动自组织网络的服务发现协议。本文首先研究了广泛应用于资源分享的对等网络与移动自组织网络的相似点与不同点，并探讨了将对等网络中用于资源定位的分布式哈希表的思想用于移动自组织网络中的服务发现场景的可能性。本文重点考虑了移动自组织网络的特点，将用于对等网络的分布式哈希表系统针对移动自组织网络的特点加以修改，并采用了服务缓存、网络数据包“偷听”等措施对协议加以优化，还采用了一种分簇的方法进一步减少了服务发现协议对网络的压力。在本文的第五章，我们在无线网络模拟器 GloMoSim 上实现了所提出的协议，并通过仿真实验对其性能进行了评估。仿真实验的结果表明本协议能够较好地在移动自组织网络的环境中完成服务发现的任务，并且其最大的优点在于协议为了实现服务发布和发现而引入的网络流量很小。

作为未来工作，本文所提出的服务发现协议仍然需要进一步的改进。由于移动自组织网络中节点对于网络全局信息的缺乏，本文提出的协议的服务发现成功率略低于采用洪泛策略的服务发现方法。因此我们还需要一种更为可靠和有效的分布式哈希表初始化策略来使节点获得更多更有效的信息。另外，服务资源的根节点突然离开网络的对网络中服务信息一致性还需要进一步地讨论。

参考文献

[1]. **Arnold K. and Scheifler, R. and Waldo, J. and O'Sullivan, B. and Wollrath, A.** Jini Specification [Book]. - [s.l.] : Addison-Wesley Longman Publishing Co., Inc., 1999.

[2]. **Basagni S.** Distributed clustering for ad hoc networks [Conference] // Parallel Architectures, Algorithms, and Networks, 1999.(I-SPAN'99) Proceedings. Fourth International Symposium on. - 1999. - pp. 310–315.

[3]. **Caesar M. and Castro, M. and Nightingale, E.B. and O'Shea, G. and Rowstron, A.** Virtual ring routing: network routing inspired by DHTs [Conference] // ACM SIGCOMM Computer Communication Review. - [s.l.] : ACM, 2006. - pp. 351--362.

[4]. **Cramer C. and Stanze, O. and Weniger, K. and Zitterbart, M.** Demand-driven clustering in manets [Conference] // International Workshop on Mobile Ad Hoc Networks and Interoperability Issues (MANETII04). - Las Vegas, USA : [s.n.], 2004.

[5]. **Das S. and Perkins, C. and Royer, E.** Ad hoc on demand distance vector (AODV) routing [Journal] // Mobile Ad-hoc Network (MANET) Working Group, IETF. - 2002.

[6]. **Helal S. and Desai, N. and Verma, V. and Lee, C.** Konark-a service discovery and delivery protocol for ad-hoc networks [Conference] // Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE. - 2003. - pp. 2107–2113.

[7]. **Hu Y.C. and Das, S.M. and Pucha, H.** Exploiting the synergy between peer-to-peer and mobile ad hoc networks [Journal] // Hot-OS IX. - 2003.

[8]. **Johnson D.B. and Maltz, D.A.** Dynamic source routing in ad hoc wireless networks [Journal] // Mobile computing. - 1996. - pp. 153--181.

[9]. **Kozat U. C and Tassiulas, L.** Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues [Journal] // Ad Hoc Networks. - 2004. - pp. 23–44.

[10]. **Kwon T.J. and Gerla, M. and Varma, V.K. and Barton, M. and Hsing, T.R.** Efficient flooding with passive clustering-an overhead-free selective forward mechanism for ad hoc/sensor networks [Journal] // Proceedings of the IEEE. - 2003. - pp. 1210--1220.

[11]. **Lin C.R. and Gerla, M.** Adaptive clustering for mobile wireless networks [Journal] // Selected Areas in Communications, IEEE Journal on. - 1997. - pp. 1265--1275.

[12]. **Maymounkov P. and Mazieres, D.** Kademlia: A peer-to-peer information system based on the xor metric [Journal] // Peer-to-Peer Systems. - 2002. - pp. 53--65.

[13]. Napster [Online]. - http://www.napster.com/.

[14]. OASIS Web Services Dynamic Discovery (WS-Discovery) Version 1.1 [Online]. - http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html.

[15]. **Pei G. and Gerla, M. and Hong, X. and Chiang, C. C** A wireless hierarchical routing protocol with group mobility [Conference] // Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE. - 1999. - pp. 1538–1542.

[16]. **Plaxton C.G. and Rajaraman, R. and Richa, A.W.** Accessing nearby copies of replicated objects in a distributed environment [Journal] // Theory of Computing Systems. - 1999. - pp. 241--280.

[17]. **Pucha H. and Das, S.M. and Hu, Y.C.** Ekta: an efficient DHT substrate for distributed applications in mobile ad hoc networks [Conference] // obile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on. - [s.l.] : IEEE, 2004. - pp. 163 - 173.

[18]. **Ratnasamy S. and Francis, P. and Handley, M. and Karp, R. and Shenker, S.** A scalable content-addressable network [Journal] // ACM SIGCOMM Computer Communication Review. - [s.l.] : ACM, 2001. - pp. 161--172.

[19]. **Ratsimor O. and Chakraborty, D. and Joshi, A. and Finin, T.** Allia: Alliance-based service discovery for ad-hoc environments [Conference] // Proceedings of the 2nd international workshop on Mobile commerce. - 2002. - pp. 1–9.

[20]. RFC 3224 - Vendor Extensions for Service Location Protocol, Version 2 [Online]. - http://tools.ietf.org/html/rfc3224.

[21]. RFC 4632 - Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan [Online]. - http://tools.ietf.org/html/rfc4632.

[22]. **Rowstron A. and Druschel, P.** Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems [Conference] // Middleware 2001. - [s.l.] : Springer, 2001. - pp. 329--350.

[23]. Standards - OASIS [Online]. - http://www.oasis-open.org/standards#uddiv3.0.2.

[24]. **Stoica I. and Morris, R. and Karger, D. and Kaashoek, M.F. and Balakrishnan, H.** Chord: A scalable peer-to-peer lookup service for internet applications [Conference] // ACM SIGCOMM Computer Communication Review. - 2001. - pp. 149–160.

[25]. **Winter R. and Zahn, T. and Schiller, J.** Random landmarking in mobile, topology-aware peer-to-peer networks [Journal] // Distributed Computing Systems, 2004. FTDCS 2004. Proceedings. 10th IEEE International Workshop on Future Trends of. - 2004. - pp. 319–324.

[26]. **Yu A. and Vuong, S.T.** A DHT-based hierarchical overlay for Peer-to-Peer MMOGs over MANETs [Journal] // Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International. - 2011. - pp. 1475-1480.

[27]. **Yu J.Y. and Chong, P.H.J.** A survey of clustering schemes for mobile ad hoc networks [Journal] // Communications Surveys & Tutorials, IEEE. - [s.l.] : IEEE, 2005. - 1 : Vol. 7.

[28]. **Zahn T. and Schiller, J.** MADPastry: A DHT substrate for practicably sized MANETs [Conference] // Proc. of ASWN. - 2005.

[29]. **Zeng X. and Bagrodia, R. and Gerla, M.** GloMoSim: a library for parallel simulation of large-scale wireless networks [Conference] // Parallel and Distributed Simulation, 1998. PADS 98. Proceedings. Twelfth Workshop on. - 1998. - pp. 154-161.

[30]. **Zhao B.Y. and Kubiatowicz, J. and Joseph, A.D. and others** Tapestry: An infrastructure for fault-tolerant wide-area location and routing [Journal]. - [s.l.] : Citeseer, 2001.

[31]. **Zhou X. and Ge, Y. and Chen, X. and Jing, Y. and Sun, W.** Smf: A novel lightweight reliable service discovery approach in manet [Conference] // Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on. - [s.l.] : IEEE, 2011. - pp. 1-5.

本科学习期间参加项目和发表论文情况

1. 本科期间在国际会议发表 EI 索引论文 2 篇

   [1]. Xi Zhou; Yifan Ge; Xuxu Chen; Yinan Jing; Weiwei Sun; , "A Distributed Cache Based Reliable Service Execution and Recovery Approach in MANETs," *Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific* , vol., no., pp.298-305, 12-15 Dec. 2011

   [2]. Xi Zhou; Yifan Ge; Xuxu Chen; Yinan Jing; Weiwei Sun; , "SMF: A Novel Lightweight Reliable Service Discovery Approach in MANET," *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on* , vol., no., pp.1-5, 23-25 Sept. 2011

2. 申请并完成著政项目：“移动自组织网络环境下一种快速可靠的服务发现方法”
3. 参与移动数据管理实验室 863 课题：“自组织网络中的服务合成研究”

致谢

本科的生活已经接近尾声，在这四年期间有近三年是在移动数据管理实验室度过的。借此机会，首先表示对实验室主任同时也是毕业论文导师的孙未未博士的感谢。孙老师长期以来的指导、关心和督促是我在实验室期间工作的动力，严谨治学和科学研究的精神也是我学习的榜样。同样要感谢一直给予我指导和支持的荆一楠老师，荆老师丰富的专业知识以及对于我工作的支持和督促是我完成论文的保障。

还要感谢移动数据管理实验室的朱良、陈坤杰、史元界、陈楚南、刘鹏、张健、陈依娇、吴晶晶等学长和学姐们，以及陈旭旭、屠川川、陈翀、刘未末等同学们，还有诸位学弟学妹们，你们各方面的帮助和陪伴使我在实验室的生活变得精彩。特别还要感谢已经离开实验室的毛鼎鼎和陈炜于学长，你们的成果是我工作的坚实基础。

还要感谢软件学院的陈荣华老师、周曦老师以及叶雅珍老师，你们的帮助是我顺利完成各项工作的保证。

另外，还要感谢曾给我授过课的每一位老师，你们的专业知识令我受益匪浅。